Bash Script

CIRC Summer School 2015

Baowei Liu

UNIVERSITY *of* ROCHESTER

Exit Status of Commands

A successful command returns 0 (shell true) while an unsuccessful command returns non-zero (shell false)

- Use echo $? To check the exit status
- true and false commands
- [[…]]; echo $?

Conditional Expression

```
 if command
 then
 …
fi
```

Conditional Executions & Arguments

- Command 1 && Command 2
- Command 1 || Command 2

Brace Expansion

- Brace expansion is used to generate an list.
- {string1, string2, …,stringN}

  space not allowed between braces!!!
- Range{<start>..<end>}: {1..20}
- Very first expansion to do !!

  {$a..$b}

Brace Expansion

- Preamble and Postscript

  a{1,3,4}b

  space is important!!!
- Combining and nesting

  {a,b,c}{1..3}

  {{a,b,c},{1..3}}
- Escaping backslash

Loop Constructs: for loop

- Basic Syntax

```
for arg in [list]
do
    …..
 done
```

- [list]:
  1. Brace Expansion (string or integer): {1..5}
  2. Command Substitution: \`ls\`
  3. Arithmetic Expansion?

for loop –Arithmetic Expansion

- Basic Syntax

```
for (( expr1; expr2; expr3 ))
do
  …
done
```

- Examples:
- White space are not important for Arithmetic Expansion

Loop Constructs –while loop

- Conditional Expression

```
while [[ conditional expression ]]
 do
    ….
 done
```

- Arithmetic Expansion

```
while (( arithmetic expression ))
 do
  …
 done
```

Loop Constructs –until loop

- Conditional Expression

```
until [[ conditional expression ]]
 do
    ….
 done
```

- Arithmetic Expansion

```
until (( arithmetic expression ))
 do
  …
 done
```

Functions

- Syntax

```
Function functname{
    commands….
  }
Function functname(){
    commands….
  }
```

- Pass Arguments
- Returning Values

Other Flow Control Constructs: case

```
case expression in
  pattern1)
          statement;;
  pattern2)
           statement;;
   ….
esac
```

;; and *